Question3

Solution :

1. Setup

Assume there are three activities $A, B\ and\ C$, $E(A,i)$ is the enjoyment if we do activity A at the $day\ i$.

2. Subproblems

Let us denote $dp(A,j)$ is the maximum total enjoyment we can get for day 0 to day j if we do activity A at the $day\ j$ and $DP(j)$ is the maximum total enjoyment we can get for day 0 to day j whatever we choose in the jth day.

Obviously, we can get $DP(j) = \max\{dp(A,j), dp(B,j), dp(C,j)\}$.

Because we are not allowed to do the same activity two days in a row, if we choose activity A at day d, we can only choose activity B and C at day $d-1$.

Hence, the subproblems are 'what's the maximum total enjoyment we can get from day 1 to yesterday if we do A or B or C yesterday'. So, the maximum total enjoyment are $dp(A,d-1), dp(B,d-1), dp(C,d-1)$. If we have solved all the subproblems:

If we choose activity A at day d, we can only choose activity B and C at day $d-1$.

So, $dp(A,d) = \max\{dp(B,d-1)+E(A,d), dp(C,d-1)+E(A,d)\}$.

If we choose activity B at day d, we can only choose activity A and C at day $d-1$.

So, $dp(B,d) = \max\{dp(A,d-1)+E(B,d), dp(C,d-1)+E(B,d)\}$

If we choose activity C at day d, we can only choose activity A and B at day $d-1$.

So, $dp(C,d) = \max\{dp(B,d-1)+E(C,d), dp(A,d-1)+E(C,d)\}$

3. **Build-up order**

Solve the subproblems in the order $dp(A,1), dp(B,1), dp(C,1), dp(A,2)$ $dp(B,2), dp(C,2), \dots, dp(A,N), dp(B,N), dp(C,N)$

4. **Recursion**

Assume we have solve all the subproblem for $t < m$, it means that we get $dp(A,t), dp(B,t)\ and\ dp(C,t)$ for $t < m$.

$$dp(A,m) = \max\{dp(B,m-1) + E(A,m), dp(C,m-1) + E(A,m)\}$$

$$dp(B,m) = \max\{dp(A,m-1) + E(B,m), dp(C,m-1) + E(B,m)\}$$

$$dp(C,m) = \max\{dp(B,m-1) + E(C,m), dp(A,m-1) + E(C,m)\}$$

5. **base case:**

$$dp(A,1) = E(A,1);\ dp(B,1) = E(B,1); dp(C,1) = E(C,1)$$

6. **Final solution**

$$maximum\ total\ enjoyment = max\{dp(A,n), dp(B,n), dp(C,n)\}$$

7. **Time complexity**

There are $3n$ subproblems and each subproblems is $O(1)$ hence the overall time complexity of the algorithm is $O(n)$.